

1998 Northeast North America Programming Contest

Problem Number 1: The Return of the Roman Empire

Write a program that accepts a Roman numeral and converts it to decimal form.

Remember, I=1, V=5, X=10, L=50, C=100, D=500, and M=1000. Furthermore, there are the following digraphs: IV=4, IX=9, XL=40, XC=90, CD=400, CM=900.

The program should reject improperly formed numerals.

Sample Input:

MCMXCVIII

Program Output:

1998

Sample Input:

CCM

Program Output:

This is not a valid number

Problem 2: DNA Sequencing

A DNA molecule consists of two strands that wrap around each other to resemble a twisted ladder whose sides, made of sugar and phosphate molecules, are connected by rungs of nitrogen-containing chemicals called bases. Each strand is a linear arrangement of repeating similar units called nucleotides, which are each composed of one sugar, one phosphate, and a nitrogenous base. Four different bases are present in DNA: adenine (A), thymine (T), cytosine (C), and guanine (G). The particular order of the bases arranged along the sugar-phosphate backbone is called the DNA sequence; the sequence specifies the exact genetic instructions required to create a particular organism with its own unique traits.

Geneticists often compare DNA strands and are interested in finding the longest common base sequence in the two strands. Note that these strands can be represented as strings consisting of the letters a, t, c, and g. So, the longest common sequence in the two strands *atgc* and *tga* is *tg*. It is entirely possible that two different common sequences exist that are the same length and are the longest possible common sequences. For example in the strands *atgc* and *gctg*, the longest common sequences are *gc* and *tg*.

Write a program that accepts as input two strings representing DNA strands, and prints as output the longest common sequence(s) in lexicographical order.

Sample Input:

```
atgc
tgc
```

Sample Output:

```
tg
```

Sample Input:

```
atgc
gctg
```

Sample Output:

```
gc
tg
```

Problem 3: Transform those strings

Given below is a set of rules that can be used to transform one string into another:

1. 't' → 'i' // change 't' to 'i'
2. { 'a' - 'z' } → 'm' // change 'a', or 'b', or 'c', ... , or 'z' to 'm'
3. 's' → 'i'
4. 'a' → 'aa'
5. { 'a' - 'z' } → "" // delete 'a', or 'b', or 'c', ... , or 'z'
6. 't' → 'r'
7. { 'a' - 'g' } → 'h'
8. 'o' → 'a'
9. 't' → 's'
10. { 't' - 'z' } → 'u'

Sets are always given as a range, i.e. { 'a' - 'c' } is the set with the characters { 'a', 'b', 'c' }

Write a program that reads two strings, and prints the steps required to transform the first string into the second. The steps in the transformation must be printed in the form:

rule: origString → newString

Note that several transformations may exist, your program only needs to find one. It is also possible that the first string cannot be transformed into the second. In this case, your program should simply output the word "no"

Sample input:

thinner
rih

Sample output:

6: thinner -> rhinner
5: rhinner -> rinner
7: rinner -> rinnhr
5: rinnhr -> rinhr
5: rinhr -> rihr
5: rihr -> rih

Sample input:

a
x

Sample output:

no

Problem 4: We Ship Cheap

The We-Ship-Cheap package shipping company is always interested in reducing their costs. They have decided that a computerized shipping route planner that determines the shortest shipping route between two cities would speed package delivery. We-Ship-Cheap services a number of different cities. They have established direct shipping links between pairs of cities. It is somewhat unusual that they have been able to create all the direct links with exactly the same distance. Unfortunately, since not every pair of cities is connected by a direct link, the shortest shipping route often involves travel through multiple intermediate cities.

Write a program that given a collection of cities and links between them, and a shipping request, prints out the shortest shipping route for the given shipping request.

The program takes as input a variable number of lines, each consisting of exactly two cities (identified by a two-letter code and separated by a space). Each line identifies a bi-directional link that exists between the two cities. Following the link information will be a single shipping request that identifies a source and destination city. The program will produce as output a shortest shipping route (note that multiple minimum length routes may exist) between the two cities. If no route exists, the program should output “No route”

The first line of input will consist of a single integer that represents the number of links given in the input. The last line will contain the source and destination cities for which you are to find the minimal route. You may assume that the input is valid and consistent.

Sample input:

```
3
JV PT
KA PT
KA HP
JV HP
```

Sample output:

```
JV PT
PT KA
KA HP
```

Sample input:

```
2
JV PT
KA HP
JV HP
```

Sample output:

```
No route
```

Problem 5: Fibinary Numbers

The standard interpretation of the binary number 1010 is $8 + 2 = 10$. An alternate way to view the sequence “1010” is to use Fibonacci numbers as bases instead of powers of two. For this problem, the terms of the Fibonacci sequence are:

1, 2, 3, 5, 8, 13, 21, ...

Where each term is the sum of the two preceding terms (note that there is only one 1 in the sequence as defined here). Using this scheme, the sequence “1010” could be interpreted as $1 \bullet 5 + 0 \bullet 0 + 1 \bullet 2 + 0 \bullet 1 = 7$. This representation is called a Fibinary number.

Note that there is not always a unique Fibinary representation of every number. For example the number 10 could be represented as either $8 + 2$ (10010) or as $5 + 3 + 2$ (1110). To make the Fibinary representations unique, larger Fibonacci terms must always be used whenever possible (i.e. disallow 2 adjacent 1's). Applying this rule to the number 10, means that 10 would be represented as $8+2$ (10010).

Write a program that takes two valid Fibinary numbers and prints the sum in Fibinary form. You may assume that any arithmetic you need to perform will fit in a 32-bit integer.

Sample Input:

10010
1

Sample Output:

10100

Sample Input:

10000
1000

Sample Output:

100000

Sample Input:

10000
10000

Sample Output:

100100

Problem 6: Pentominos

A pentomino consists of five equal-sized squares attached edge-to-edge to form some shape. There are twelve possible pentominos that can be formed in this way (plus their reflections and rotations). A few pentominos are shown below:

XXXXX

XXXX
X

XXXX
X

X
XXX
X

Something that has been keeping people occupied since the late 50's is to find a way to form a rectangle of a given size using the 12 different pentominos

Write a program that reads 4 lines containing two numbers between 1 and 8. These numbers represent 4 different cells in an 8x8 square. The program must determine if it is possible to cover the remaining 60 cells using the 12 different types of pentominos. The solution may use a given type of pentomino more than once, or not at all. The output from the program will show the square filled with pentominos. Each pentomino in the output must be drawn using a different character (you may assume that you will not need more than 52 pentominos in a given solution). The *blocked* cells should be drawn using the '*' character. Your program needs to only find one solution.

Sample Input:

3 5
4 5
5 5
6 5

Sample Output:

aaaaabcc
deeebbbc
ddde*bcc
ffde*ggg
hffi*ggj
hfii*jjj
hiikkklj
hhkkllll